



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/902,133	07/11/2001	Seiji Hayashida	211316US2	7963
22850	7590	06/01/2006	EXAMINER	
OBLON, SPIVAK, MCCLELLAND, MAIER & NEUSTADT, P.C. 1940 DUKE STREET ALEXANDRIA, VA 22314			RAMPURIA, SATISH	
			ART UNIT	PAPER NUMBER
			2191	

DATE MAILED: 06/01/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No. 09/902,133	Applicant(s) HAYASHIDA, SEIJI	
	Examiner Satish S. Rampuria	Art Unit 2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 10 March 2006.
- 2a) ☐ This action is FINAL. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-20 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-20 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

Response to Amendment

1. This action is in response to the RCE received on March 10, 2006.
2. The rejections under 35 U.S.C. §112 first paragraph to claims 1, 4 and 7 is withdrawn in view of Applicant's amendment.
3. Claims amended by the Applicant: 1, 2, 4, 5, 7, 8, 9, 14, 16, 17, 19 and 20.
4. Claims pending in the application: 1-20.
5. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on March 10, 2006 has been entered.

Response to Arguments

6. Applicant's arguments with respect to claims have been considered but they are not persuasive.

In the remarks, the applicant has argued that:

- (i) Neither Caron nor Santhanam describe or suggest "an intrinsics function information database into which a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics function are stored as intrinsics function information." as recited in independent claims 1, 7-9, 12, 14, 16, 17, 19 and 20.

Examiner's response:

- (i) In response to Applicant's argument, both Caron and Santhanam are directed towards generating machine or executable code. More specifically, Santhanam discloses the intrinsic function information is stored in a table which is considered to be a data structure (col. 4, lines 10-15). In addition, Santhanam even discloses generation of data structures which contain information on the types of the intrinsic arguments and type of the return value (col. 13-14, lines 65-67 and lines 1-2). Further, rejection clearly points out the motivation that to make the compilation process more efficient via use of selected assembly commands at the time of compile. Applicants make general allegation. Therefore, the rejection is proper and maintained herein.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

8. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,083,282 to Caron et al., hereinafter called Caron, in view of US Patent No. 6,247,174 to Santhanam et al., hereinafter called Santhanam.

Per claims 1 and 2:

Caron discloses:

Art Unit: 2191

- A compiler system for generating object code from an input source program (col. 1, lines 21-23 “high level language... source code are then translated or compiled into the coded instructions executable by the computer”), comprising:
- a character string interpreter configured to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 “In the syntax of the programming language used... names (also known as identifiers) are strings of characters”) and col. 9, lines 21-24 “compiling process 70... performed on source code to separate the source code into various lexical constructs or tokens... programming language, including identifiers, keywords, operators, literals, and comments”);
- a syntax analyzer configured to analyzes syntax of said tokens (col. 9, lines 24-27 “Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like”), to judge as to whether or not a definition of an intrinsic function is included in a combination of said tokens, to find a reserved pre-processing instruction in the combination of said token and (col. 9, lines 35-37 “compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code”), if found, to add said instruction attribute information of said intrinsic function described in said pre-processing instruction to the definition of the intrinsic functions in said database (col. 9, lines 55-67 to col. 11, line 25 “the compiler locates a name of a program element identical to the name reference by searching the namespaces of various procedures, modules, and projects of the source code in a particular order... The name is then added to the

namespace of the current procedure or module in which it appears. However, if implicit declaration of variables is not permitted, an error is instead generated by the compiler”).

Caron does not explicitly disclose whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens, an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information; a code generator configured to develop an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information, and to convert said developed source program either to machine language or to an intermediate code.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 “Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct”) and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 “The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic

into a low-level machine instruction”) and a code generator (col. 6, lines 30 “Code generator 105”) configured to develop an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 “programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s”), and to convert said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 “Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 3:

The rejection of claim 1 is incorporated, and further, Caron does not explicitly disclose intrinsic function definition includes a dummy argument type and identification name.

However, Santhanam in an analogous computer system discloses intrinsic function definition includes a dummy argument type and identification name (col. 4, lines 10-13 “The table contains one entry for each intrinsic... name... data type... argument... value”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of containing intrinsic function information as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to include the intrinsic function definition to improve the run time performance of the library function invocation as suggested by Santhanam (col. 2, lines 60-65).

Per claim 4 and 5:

Caron discloses:

- A compiler system for generating object code from an input source code program (col. 1, lines 21-23 “high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer”), comprising:
- a character string interpreter configured to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 “In the syntax of the programming language used... names (also known as identifiers) are strings of characters”) and col. 9, lines 21-24 “In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments”);
- a syntax analyzer, configured to analyze syntax of said tokens (col. 9, lines 24-27 “Syntax analysis 74 also is performed to separate the source code into syntax structures, such as declaration statements, loop statements, expressions, and the like”), to judge as

to whether or not a definition of an intrinsic function is included in a combination of said tokens, to find a reserved pre-processing instruction in the combination of said token and (col. 9, lines 35-37 “compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code”), if found, to add said instruction attribute information of said intrinsic function described in said pre-processing instruction to the definition of the intrinsic functions in said database (col. 9, lines 55-67 to col. 11, line 25 “the compiler locates a name of a program element identical to the name reference by searching the namespaces of various procedures, modules, and projects of the source code in a particular order... The name is then added to the namespace of the current procedure or module in which it appears. However, if implicit declaration of variables is not permitted, an error is instead generated by the compiler”).

Caron does not explicitly disclose whether or not a definition of an intrinsic function and an instruction attribute information characterizing an instruction coded in intrinsic functions is included in a combination of said tokens, an intrinsic function information database into which a definition of said intrinsic function and said instruction attribute information are stored as intrinsic function information, and a code generator configured to develop an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information, and to convert said developed source program either to machine language or to an intermediate code.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsic function and an instruction attribute information characterizing an

Art Unit: 2191

instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 “Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates... code containing embedded machine instructions... correct when... incorporated... source code 101 in the same way... a front end verifies... function invocation... correct”) and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 “The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction”) and a code generator (col. 6, lines 30 “Code generator 105”) configured to develop an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 “programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s”), and to convert said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 “Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator

Art Unit: 2191

to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claim 6:

The rejection of claim 5 is incorporated, and further, Caron does not explicitly disclose intrinsic function definition includes a dummy argument type and identification name.

However, Santhanam in an analogous computer system discloses intrinsic function definition includes a dummy argument type and identification name (col. 4, lines 10-13 “The table contains one entry for each intrinsic... name... data type... argument... value”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of containing intrinsic function information as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to include the intrinsic function definition to improve the run time performance of the library function invocation as suggested by Santhanam (col. 2, lines 60-65).

Per claim 7:

Caron discloses:

- A compiler system for generating object code from an input source program (col. 1, lines 21-23 “high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer”), comprising:

- a character string interpreter configured to divide instructions coded within an input source program into tokens (col. 7, lines 20-24 “In the syntax of the programming language used... names (also known as identifiers) are strings of characters”) and col. 9, lines 21-24 “In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments”);
- a syntax analyzer configured to analyzes syntax of said tokens (col. 9, lines 24-27 “Syntax analysis 74 also is performed to separate the source code into syntax structures, such as declaration statements, loop statements, expressions, and the like”) to judge as to whether or not a definition of an intrinsic function is included in a combination of said tokens, to find a reserved pre-processing instruction in the combination of said token and (col. 9, lines 35-37 “compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code”), if found, to add said instruction attribute information of said intrinsic function described in said pre-processing instruction to the definition of the intrinsic functions in said database (col. 9, lines 55-67 to col. 11, line 25 “the compiler locates a name of a program element identical to the name reference by searching the namespaces of various procedures, modules, and projects of the source code in a particular order... The name is then added to the namespace of the current procedure or module in which it appears. However, if implicit declaration of variables is not permitted, an error is instead generated by the compiler”);

Caron does not explicitly disclose whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens, an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information, and a code generator configured to develop an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information, and to convert said developed source program either to machine language or to an intermediate code and wherein said intrinsics function information includes a function declaration statement, to which is added a prescribed identifier indicating an intrinsics function, dummy argument information, and said instruction attribute information.

However, Santhanam in an analogous computer system discloses, whether or not a definition of an intrinsics function and an instruction attribute information characterizing an instruction coded in intrinsics functions is included in a combination of said tokens (col. 12, lines 46-53 “Table-driven... processing enables... feature... automatic syntax parsing and semantics checking... inline assembly code by FE 102... feature validates that code containing embedded machine instructions is... correct... when... incorporated... source code 101 in the same way... a front end verifies.... function invocation...”) and an intrinsics function information database into which a definition of said intrinsics function and said instruction attribute information are stored as intrinsics function information (col. 4, lines 10-15 “The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction”) and a code generator (col. 6, lines 30 “Code generator 105”) configured to develop

an instruction that calls an intrinsic function within said source program by referring to said intrinsic function information (col. 7, lines 1-3 “programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s”), and to convert said developed source program either to machine language or to an intermediate code (col. 6, lines 30-33 “Code generator 105 then translates high-level intermediate representation 103 into low-level intermediate representation 106”) and wherein said intrinsic function information includes a function declaration statement, to which is added a prescribed identifier indicating an intrinsic function, dummy argument information, and said instruction attribute information (col. 4, lines 10-13 “The table contains one entry for each intrinsic... name... data type... argument... value”).

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of checking the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to check the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Per claims 8, 9, 10, and 11:

Caron discloses:

- A method for compiling which generates object code from an input source program (col. 1, lines 21-23 “high level language statements of the source code are then translated or compiled into the coded instructions executable by the computer”), comprising:
- dividing instructions coded within an input source program into tokens (col. 7, lines 20-24 “In the syntax of the programming language used... names (also known as identifiers) are strings of characters”) and col. 9, lines 21-24 “In the compiling process 70, lexical analysis 72 is first performed on source code to separate the source code into various lexical constructs or tokens of the programming language, including identifiers, keywords, operators, literals, and comments”);
- analyzing syntax of said tokens (col. 9, lines 24-27 “Syntax analysis 74 also is performed to separate the source code into syntax structures... declaration statements, loop statements, expressions, and the like”), and judging as to whether or not a definition of an intrinsic function and its kind of operand is included in a combination of said tokens; finding a reserved pre-processing instruction in the combination of said token and (col. 9, lines 35-37 “compiler determines which names reside in the various namespaces of the procedures, modules, and projects in the source code”), if found, adding said instruction attribute information of said intrinsic function described in said pre-processing instruction to the definition of the intrinsic function (col. 9, lines 55-67 to col. 11, line 25 “the compiler locates a name of a program element identical to the name reference by searching the namespaces of various procedures, modules, and projects of the source code in a particular order... The name is then added to the namespace of the current

procedure or module in which it appears. However, if implicit declaration of variables is not permitted, an error is instead generated by the compiler”).

Caron does not explicitly disclose storing a definition of an intrinsics function into an intrinsics function information database; storing instruction attribute information characterizing an instruction coded by an intrinsics function into said intrinsics function information database; developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information database, and converting said developed source program either to machine language or to intermediate code.

However, Santhanam in an analogous computer system discloses storing a definition of an intrinsics function into an intrinsics function information database (col. 14, lines 54-57 “information ... identity... of... intrinsic... described... is stored in the same data structures described in A”) and storing instruction attribute information characterizing an instruction coded by an intrinsics function into said intrinsics function information database (col. 4, lines 10-15 “The table contains... entry... each intrinsic... the entry describing characteristics... intrinsic... name... data types... opcode arguments... return value (if any)... relevant to translating the intrinsic into a low-level machine instruction”) and developing an instruction that calls an intrinsics function within said source program by referring to said intrinsics function information database (col. 7, lines 1-3 “programs that make intrinsic calls... refer... and incorporate... types of files into source code 201.sub.s”), and converting said developed source program either to machine language or to intermediate code (col. 6, lines 30-33 “Code generator 105 then

translates high-level intermediate representation 103 into low-level intermediate representation 106").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of storing the definitions of intrinsic functions, an intrinsic functions database, and a code generator as taught by Santhanam into the method of tokenizing the instructions from the source code as taught by Caron. The modification would be obvious because of one of ordinary skill in the art would be motivated to store the definitions of intrinsic functions, an intrinsic functions database, and a code generator to make the compilation process more efficient via use selected assembly commands at the time of compile as suggested by Santhanam (col. 2, lines 11-65).

Claims 12 and 17 are the computer program product claim corresponding to system claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claims 13 and 18 are the computer program product claim corresponding to method claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Claims 14 and 19 are the computer program product claim corresponding to method claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above.

Claim 15 is the computer program product claim corresponding to method claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Claims 16 and 20 are the computer program product claim corresponding to method claims 1 and 3 respectively, and rejected under the same rational set forth in connection with the rejection of claim 1 and 3 respectively, above.

Conclusion

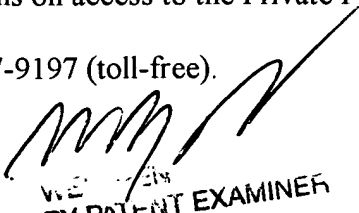
9. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Satish S. Rampuria** whose telephone number is **(571) 272-3732**. The examiner can normally be reached on **8:30 am to 5:00 pm** Monday to Friday except every other Friday and federal holidays. Any inquiry of a general nature or relating to the status of this application should be directed to the **TC 2100 Group receptionist: 571-272-2100**.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Wei Y. Zhen** can be reached on **(571) 272-3708**. The fax phone number for the organization where this application or proceeding is assigned is **571-273-8300**.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner/Software Engineer
Art Unit 2191


WEI Y. ZHEN
SUPERVISORY PATENT EXAMINER